D-ITET, D-MATL
Dr. R. Käppeli
B. Pratyuksh



# Serie 1

**1.** Diese Aufgabe ist als Einführung zu MATLAB zu verstehen. Wir empfehlen diese Aufgabe insbesondere denjenigen unter Ihnen, die noch nie programmiert haben. Falls Sie Erfahrung mit anderen Programmiersprachen haben, ist diese Aufgabe nützlich, um sich mit der Syntax von MATLAB vertraut zu machen. Falls Sie bereits MATLAB benutzt haben, empfehlen wir die Aufgabe kurz durchzuschauen, um sicherzustellen, dass Sie mit allen hier behandelten Konzepten vertraut sind.

1. Werten Sie die Matrix-Vektor Multiplikation $Ab$ und die Matrix-Matrix Multiplikation $AB$ mittels MATLAB für die folgenden Matrizen und den folgenden Vektor aus:

$$A = \begin{pmatrix} 2 & 3 & 1 & 4 \\ -1 & -5 & 0 & 2 \\ 4 & 4 & 0 & 1 \\ 1 & -1 & 7 & 6 \end{pmatrix}, \; B = \begin{pmatrix} 1 & 2 & 3 & -3 \\ 0 & 3 & 0 & 2 \\ -1 & -1 & 2 & 3 \\ 4 & 3 & 2 & -1 \end{pmatrix}, \; b = \begin{pmatrix} 3 \\ 2 \\ 0 \\ 1 \end{pmatrix}.$$

   *Hinweis:* Die Lösung ist (zur Kontrolle für Sie)

$$Ab = \begin{pmatrix} 16 \\ -11 \\ 21 \\ 7 \end{pmatrix}, \; AB = \begin{pmatrix} 17 & 24 & 16 & -1 \\ 7 & -11 & 1 & -9 \\ 8 & 23 & 14 & -5 \\ 18 & 10 & 29 & 10 \end{pmatrix}.$$

2. Berechnen Sie mithilfe von MATLAB die Lösung $x$ des Systems $Ax = b$ für $A$ und $b$ vom Teil 1.
   *Hinweis:* Die Lösung ist

$$x = A^{-1}b = \begin{pmatrix} -0.3360 \\ 0.0787 \\ -0.6798 \\ 1.0289 \end{pmatrix}.$$

3. Gegeben sei $N \in \mathbb{N}$, schreiben Sie eine MATLAB Funktion, die die Summe $\sum_{k=1}^{N} k^2$ in Abhängigkeit von $N$ korrekt berechnet.
   a) Realisieren Sie die Funktion mithilfe des Befehls `for`.

**Bitte wenden!**

b) Realisieren Sie die Funktion mithilfe der Vektorschreibweise.
*Hinweis:* Sie können einen Zeilenvektor `x` mit Einträgen 1,2,...,N erzeugen mithilfe des Befehls `x = 1:N;`

4. Gegeben sei ein Vektor $v$ mit positiven Einträgen und eine Zahl $\texttt{lim} \in \mathbb{N}$. Die MATLAB Funktion `vektorindex(v,lim)` addiert sukzessive die Einträge des Vektors $v$ bis der Wert der Summe mindestens so groß ist wie die gegebene Zahl `lim`. Der Rückgabewert `ind` ist der zuletzt aufaddierte Index (sodass $\sum_{k=1}^{\texttt{ind}} v_k \geq \texttt{lim}$ und $\sum_{k=1}^{\texttt{ind}-1} v_k < \texttt{lim}$).
   a) Überzeugen Sie sich, dass die momentane Implementierung einen Fehler hat, indem Sie mit `v=ones(20,1)` und `lim=5` testen.
   b) Benutzen Sie den MATLAB *debugger*, um den Fehler zu finden und zu korrigieren.

**2.** In dieser Aufgabe leiten wir eine der populärsten Quadraturregeln her: die Simpson-Regel

$$S[f] = \sum_{j=0}^{2} \omega_j f(x_j) \approx \int_a^b f(x)dx = I[f].$$

Die Simpson-Regel verwendet als Quadratur Stützstellen/Knoten

$$x_0 = a, \qquad x_1 = \frac{a+b}{2}, \qquad x_2 = b.$$

**a)** Berechnen Sie die Lagrange-Polynome $L_0^2(x)$, $L_1^2(x)$ und $L_2^2(x)$ passend zu den Knoten $x_0$, $x_1$ und $x_2$.

**b)** Berechnen Sie mit **a)** die Quadratur Gewichte $\omega_0$, $\omega_1$ und $\omega_2$.
*Hinweis:* $S[f] = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$.

**c)** Nun untersuchen wir den sogenannten Genauigkeitgrad, d.h. der maximale Polynom Grad welche eine Quadraturregel exakt integrieren kann, der Simpson-Regel. Ohne Beschränkung der Allgemeinheit setzen wir $a = -1$ und $b = 1$ (jedes 1-dimensionale Intervall kann auf $[-1, 1]$ transformiert werden). Berechnen Sie

$$S[1],\ S[x],\ S\left[x^2\right],\ S\left[x^3\right] \text{ und } S\left[x^4\right],$$

und vergleichen Sie diese mit den exakten Integralen

$$I[1] = 2,\ I[x] = 0,\ I\left[x^2\right] = \frac{2}{3},\ I\left[x^3\right] = 0 \text{ und } I\left[x^4\right] = \frac{2}{5}.$$

Was beobachten Sie?

**Siehe nächstes Blatt!**

**3.** In der Vorlesung haben wir polynomiale Interpolation zur Approximation von Funktionen kennengelernt. In dieser Aufgabe wollen wir qualitativ die Güte der Approximation untersuchen und dabei ein paar nützliche MATLAB Befehle kennenlernen. Wir betrachten die sogenannte Runge-Funktion

$$f(x) = \frac{1}{1+25x^2}, \qquad x \in [-1,1].$$

**a)** Führen Sie folgende Befehle in MATLAB aus:

```
1 n  = 7;
2 xn = linspace(-1,1,n+1);
3 yn = 1./(1 + 25*xn.^2);
4 pn = polyfit(xn,yn,n);
```

Zeile 2 erstellt $n+1 = 8$ äquidistant verteilte Punkte im Intervall $[-1,1]$. Dies sind unsere Stützstellen/Knoten. Zeile 3 berechnet die Stützwerte, d.h. $f(x)$ an der Knoten. Zeile 4 berechnet die Koeffizienten des Interpolationspolynoms.

**b)** Führen Sie folgende Befehle in MATLAB aus:

```
1 xx = linspace(-1,1,1000);
2 hold on;
3 plot(xx,1./(1+25*xx.^2),'b-');
4 plot(xx,polyval(pn,xx),'r-');
5 plot(xn,yn,'ro');
6 hold off
```

Zeile 1 erstellt 1000 äquidistant verteilte Punkte zwischen $-1$ und $1$. An diesen Punkten wollen wir die Runge-Funktion und unser Interpolationspolynom ausrechnen und plotten (Zeilen 3 und 4). Dann werden noch die Knoten und Stützwerte des Interpolationspolynoms in Zeile 5 geplottet. Zeile 2 hält den Plot fest, damit mehrere Kurven mit dem `plot` Befehl gezeichnet werden können, und Zeile 6 lässt ihn wieder los. Weitere Informationen zu den Befehlen finden Sie in der MATLAB Dokumentation.

**c)** Wiederholen Sie **a)** und **b)** mit $n = 19$. Was beobachten Sie?

**d)** Ersetzen Sie Zeile 2 in **a)** mit

```
1 xn = cos((2*(0:n)+1)/(2*(n+1))*pi)
```

und wiederholen Sie **a)** bis **c)** . Was beobachten Sie?

Dieses Beispiel sollte darauf hinweisen, dass man polynomiale Interpolation nicht blind verwenden sollte. Wir werden dies nicht weiter in dieser Vorlesung diskutieren und für weitere Informationen empfehlen wir die zusätzlich angegebene Literatur.

**Bitte wenden!**

**Abgabe:** Bis Freitag, den 02.03.2018.
Laden Sie Ihre MATLAB -Programme unter `sam-up.math.ethz.ch` hoch.
Die schriftlichen Ergebnisse sollten Sie separat in den jeweiligen Übungsgruppen abgeben.